Algorithm 1: Uniform Cost Search for a single vehicle
Plan(R_i)
Source ← S_i
Source[τ_i] ← NULL, Source[$lane_i$] ← S_i
Q ← Source, Q_2 ← NULL
while Q ≠ NULL
    Q ← Sort(Q)
    P ← Expand(Q(1))
    Q_2 ← $Q_2 \cup$ Q(1)
    Q ← (Q – Q(1)) $\cup$ P
end Q
qp ← best plan in Q_2
return qp[τ_i], qp[$lane_i$]

Algorithm 2: Expansion for a single vehicle
Expand(L_i)
A ← NULL
for v_i ← $v_{max}^{\ i}$ to 1 in small steps
    Calculate P using (2)
    for all P_a ε P
        calculate l, l_2 for P_a using (4) and (5)
            [l, l_2] ← apply small deviations to l till $connect_2(L_i, l_2)$ ε $\zeta_{dynamic}^{free}$ ∧
        connect(l_2, l) ε $\zeta_{dynamic}^{free}$, break P_a if still infeasible
        $l_2[\tau_i]$ ← $L_i[\tau_i] \cup connect_2(L_i, l_2))$
        $l_2[lane_i]$ ← L_i[lane] $\cup$ l_2
        l_2[t] ← L_i[t] + $||connect_2(L_i, l_2))||/v_i$
        A ← A $\cup$ l_2
    end P_a
    if A ≠ NULL, return A
end v_i
return NULL

Algorithm 3: Getting number of vehicles requiring independent lanes
getLaneVehicles(L_i)
A ←R_j: $e_j < e_i$, H ← R_i
$S(S_a,S_b)$ ← (x_i-len_i/2, x_i+2Δ(v_i)+len_i/2)
do
    for R_j ε A – H
        if $\tau_j(t) \otimes R_j \cap (S_a,S_b+d) \neq \varphi$
            S_2 ← ($lane_j$[a+1] – len_j/2, $lane_j$[b+2]+len_j/2)
            $S(S_a,S_b)$ ← S $\cup$ S_2
            H ← H $\cup$ R_j, A ← A - R_j
            break R_j
        end if
    end R_j
while no change in H
return H

Algorithm 4: Division of the road into lanes
getLanePositions(H, L_i)
calculate pr by (8), sort(H) by d(pr_j)
for R_j ε H
    if (11) then (12), pr_j ←l^j, continue R_j
    calculate s from (13)
    mov[R_k] ← 0 ∀ R_k ε H
    for R_k ε H in increasing deviation from R_j in H
        calculate s_k from (14)
        mov[R_k] ← max(s, s_k), s ← s – s_k
        if s≤ 0, break R_j

end R_k
if s > 0
space_per_segment ←$((s_2 - s_1) - \sum_o wid(pr_o)) / (size(H) + 1)$
for R_k ε H, l^k ← space_per_segment + [$\sum_o$(space_per_segment + $wid(pr_o)) \ \forall \ R_o < R_k$] + $wid(pr_k)/2$
break R_j
else
for $R_k<R_j$ in H, $pr'_k \leftarrow pr_k - \sum_{R_k \le R_o < R_j} mov[pr_o]$
for $R_k>R_j$ in H, $pr'_k \leftarrow pr_k + \sum_{R_j < R_o \le R_k} mov[pr_o]$
$pr'_k \leftarrow left(pr'_k) - sm - wid_k/2$
pr ← pr', l←pr
end if
end R_j
return l

Algorithm 5: Trajectory generation from the current state to the expanded state GenerateTrajectory(l, L_i)

Calculate τ^2 using (15)

Algorithm 5.1: GenerateTrajectorySelf()

$[l^i, l^i_2] \leftarrow$ apply small deviations to l^i till $connect_2(L_i, l^i_2) \ \varepsilon \ \zeta^{free}_{dynamic}(\tau_2) \wedge connect_2(l^i_2, l^i) \ \varepsilon \ \zeta^{free}_{dynamic}(\tau_2)$, return NULL if still infeasible

$\tau_i^2 \leftarrow L_i[\tau_i] \cup connect_2(L_i, l^i_2 d)$

$l_2[lane_i] \leftarrow L_i[lane_i] \cup l^i_2$

$l_2[t] \leftarrow L_i[t] + ||connect_2(L_i, l^i_2))||/v_i$

for R_j ε H

$lane_j[l][Y] \leftarrow l^j \ \forall \ a+2 \le l \le b$

Algorithm 5.2: GenerateTrajectoryElse()

$\tau^2_j \leftarrow \tau_j(t) \ \forall \ t \le lane_j[a+1][T]$

if $\tau_j(t) \notin \zeta^{free}_{dynamic}(\tau^2, t) \forall \ L_i(t) \le t \le lane_j[a+1][T]$, return NULL

for l ← a+2 to b

$lane_j[l]$ ←apply small deviations to $lane_j[l]$ till $connect_2(lane_j[l-1], lane_j[l]) \ \varepsilon \ \zeta^{free}_{dynamic}(\tau_2)$, return NULL if still infeasible

$\tau_j^2 \leftarrow L_i[\tau_j] \cup connect_2(lane_j[l-1], lane_j[l])$

$lane_j[l][T] \leftarrow lane_j[l-1][T] + || \ connect_2(lane_j[l-1], lane_j[l]) \ ||/v_j$

end l

end R_j

$l_2[\tau] \leftarrow \tau_2$

return l_2

Algorithm 6 Free State Expansion Strategy
FreeStateExpansion (L_i)
H←getLaneVehicles(L_i)
l←getLanePositions(H, L_i)
return GenerateTrajectory(l, L_i)

Algorithm 7 Vehicle Following Expansion Strategy
VehicleFollowExpansion(L_i)
Select closest R_j in Y axis, passing through P_a in the same direction
Calculate v^{red}_i from (17)
Return FreeStateExpansion(L_i) with $v_i = v^{red}_i$ and Z at $x_i+\Delta(v_i)$, Z_2 at $x_i+\Delta(v^{red}_i)$

Algorithm 8 Wait for Vehicle Expansion Strategy
WaitForVehicleExpansion(L_i)
$l_2s \leftarrow$ NULL
for R_j : R_j crosses P_a in opposite direction to R_i
calculate H
calculate τ^2 using (15)
calculate l^i_2 using (20)
calculate $lane_j[a]$ and $lane_j[b]$
$lane_j[l][Y] \leftarrow l^j[Y] \ \forall\ l \in (a, b)$
success ← GenerateTrajectoryElse()
if not success, continue R_j
tw ← $lane_j[b][T]$
for $R_k \in H - \{R_i, R_j\}$
calculate $lane_k[a]$ and $lane_k[b]$
$lane_k[l][Y] \leftarrow l^k[Y] \ \forall\ l \in (a, b)$
success ← GenerateTrajectoryElse()
if not success, continue R_j
end R_k
success ← GenerateTrajectorySelf() from L_i to l^i_2 only
$l_2[T] \leftarrow$ tw
if not success, continue R_j
$l_2s \leftarrow l_2s \cup l_2$
end R_j
return l_2s

Algorithm 9
Expand(L_i)
A ← NULL
for $v_i \leftarrow v_{max}{}^i$ to 1 in small steps
calculate P using (2)
for all $P_a \in P$
calculate H, width(P_a)
lookup H(L_i), width(L_i), strategy(L_i)
a, b, c ← NULL
if H(L_i) ≠ H
a ← FreeStateExpansion (L_i), strategy(a) ← FreeStateExpansion
b ← VehicleFollowExpansion(L_i), strategy(b) ←
VehicleFollowExpansion
else
if strategy(L_i) = FreeStateExpansion
a ← FreeStateExpansion (L_i), strategy(a) ←
FreeStateExpansion
if strategy(L_i) = VehicleFollowExpansion

b ← VehicleFollowExpansion(L_i), strategy(b) ← VehicleFollowExpansion

end if

if width(L_i)- width(P_a) > width_threshold

c ← WaitForVehicleExpansion(L), strategy(c) ← WaitForVehicleExpansion

A ← A $\cup$ a $\cup$ b $\cup$ c

end P_a

if A $\neq$ NULL, return A

end v_i

return NULL

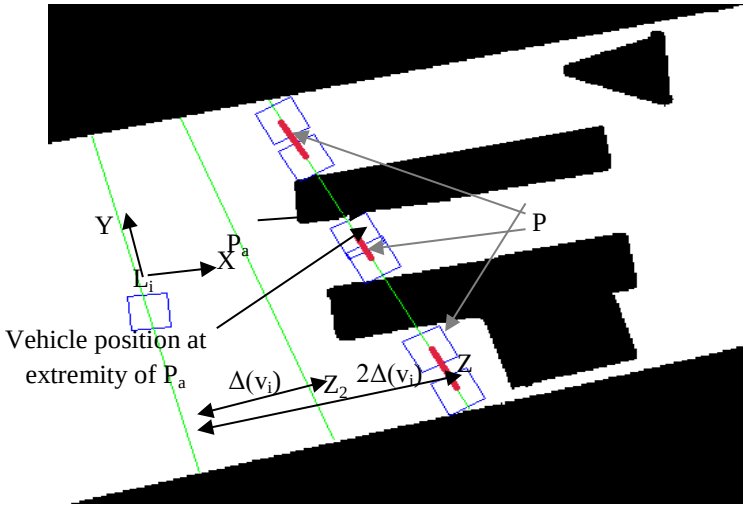

Figure 1: Obstacle analysis

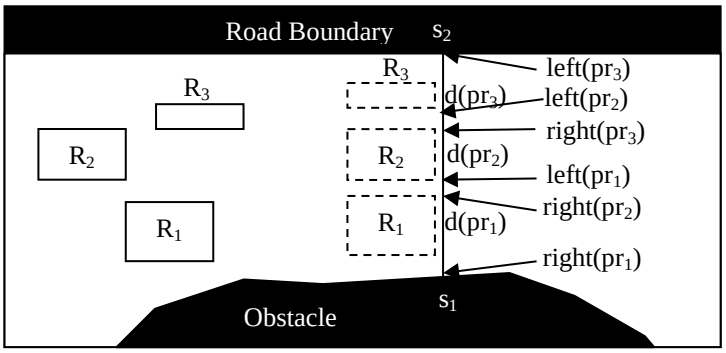

Figure 10: Distribution of lanes

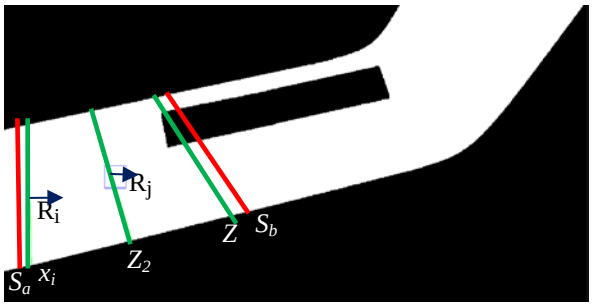

Figure 11(a): Computing lane vehicles

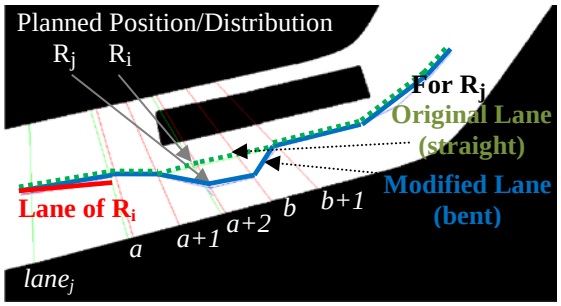

Figure 11(b): Distribution of lanes amongst vehicles (label colours correspond to line colours)

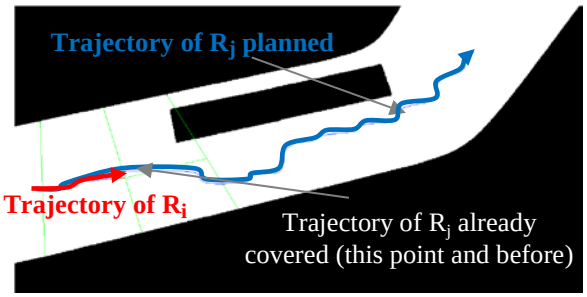

Figure 11(c): Generated trajectories for vehicles

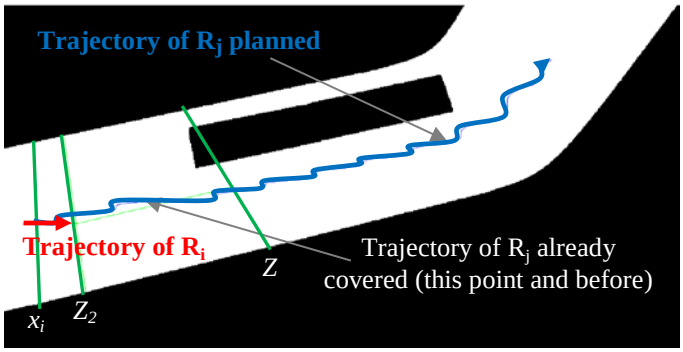

Figure 12: Trajectories of vehicles for vehicle following behaviour

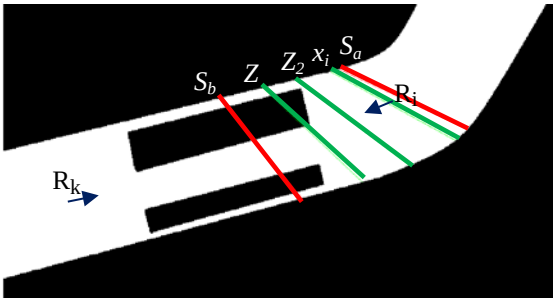

Figure 13(a): Detection of sudden narrowing of width of road

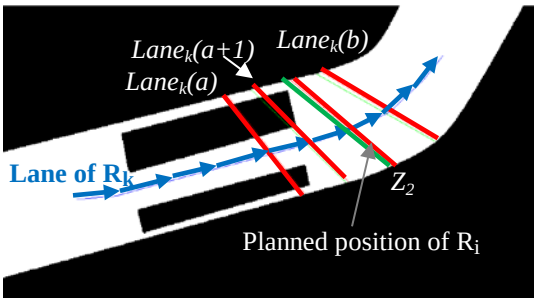

Figure 13(b): Modification of lane of R_k

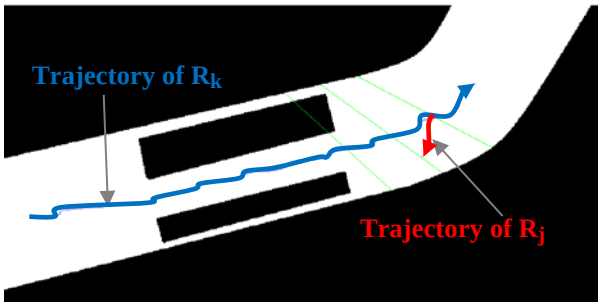

Figure 13(c): The generated trajectory for vehicles

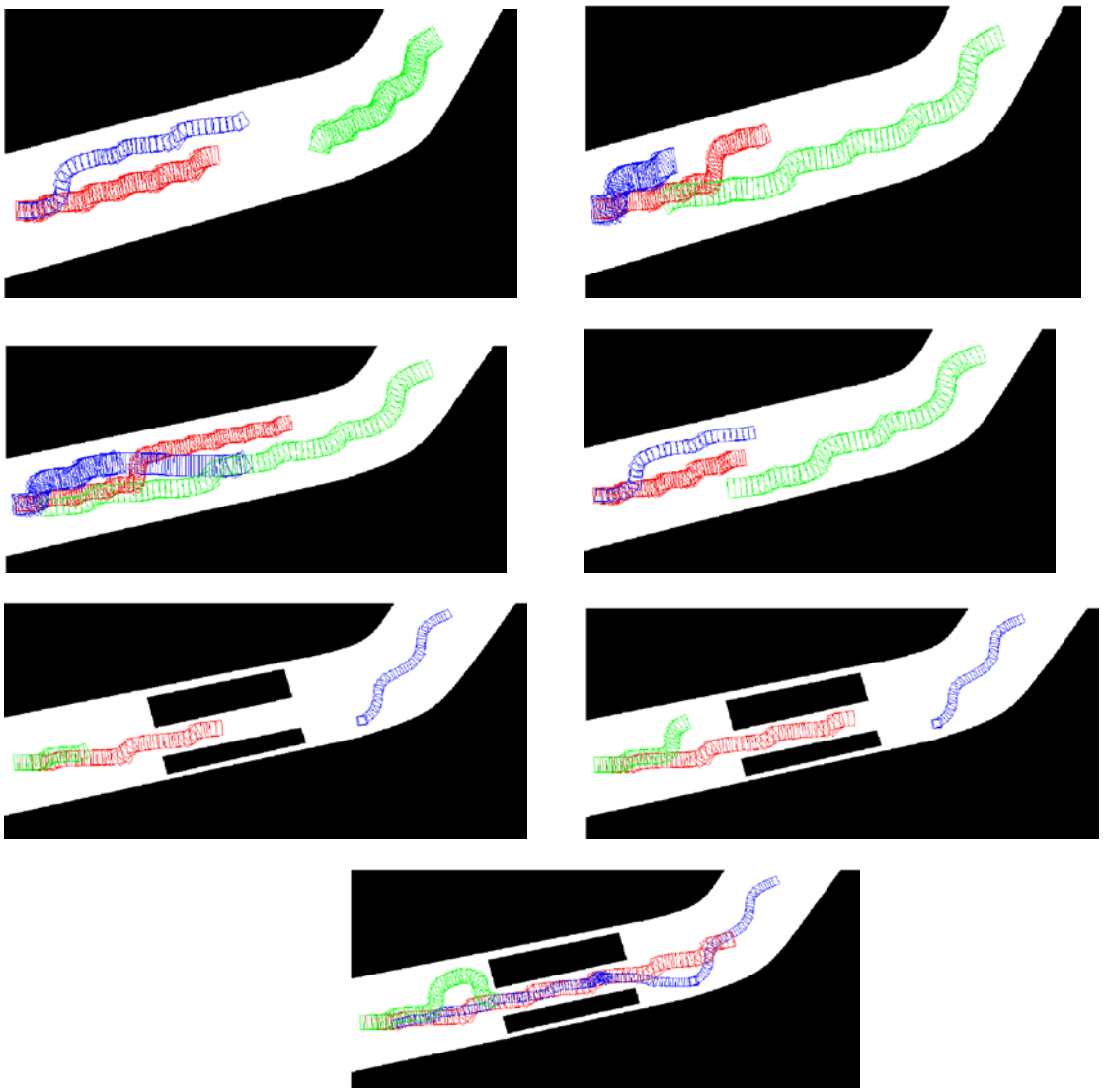

Figure 14: Experimental results with multiple vehicles without obstacles

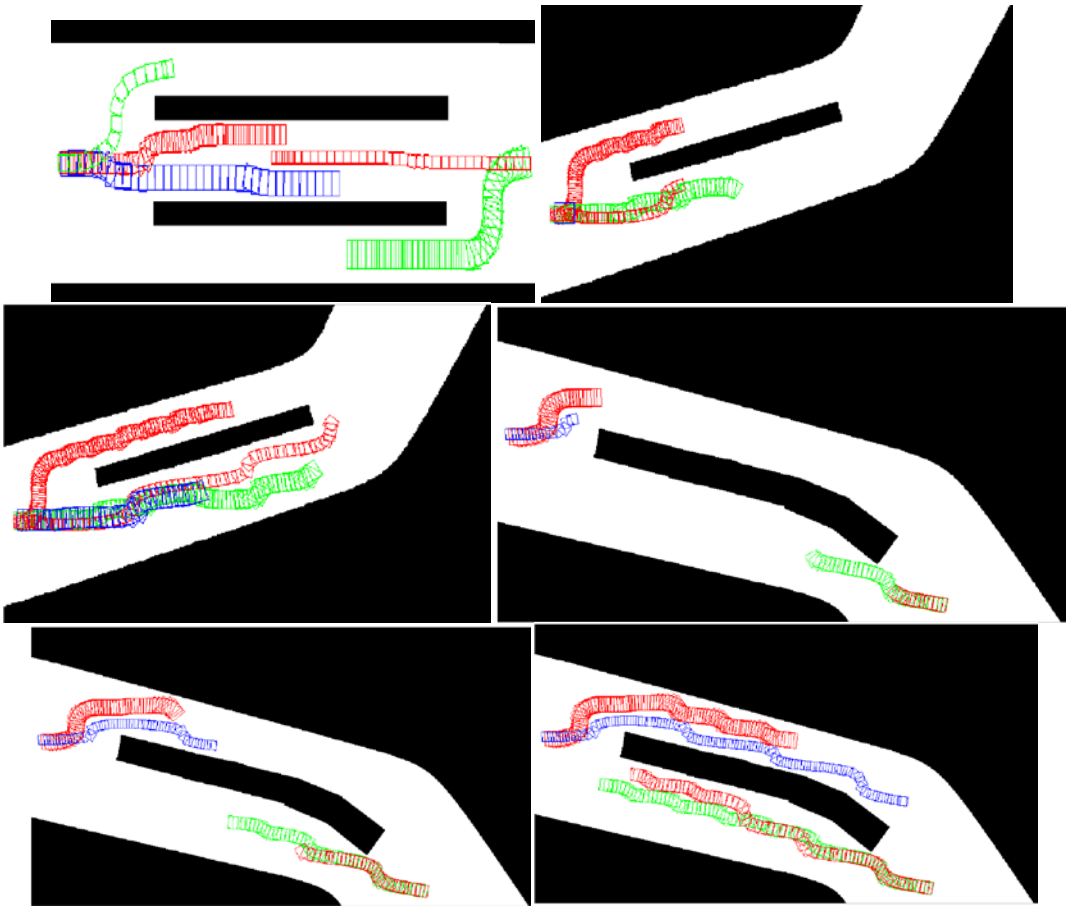

Figure 15: Experimental results with multiple vehicles in presence of obstacles

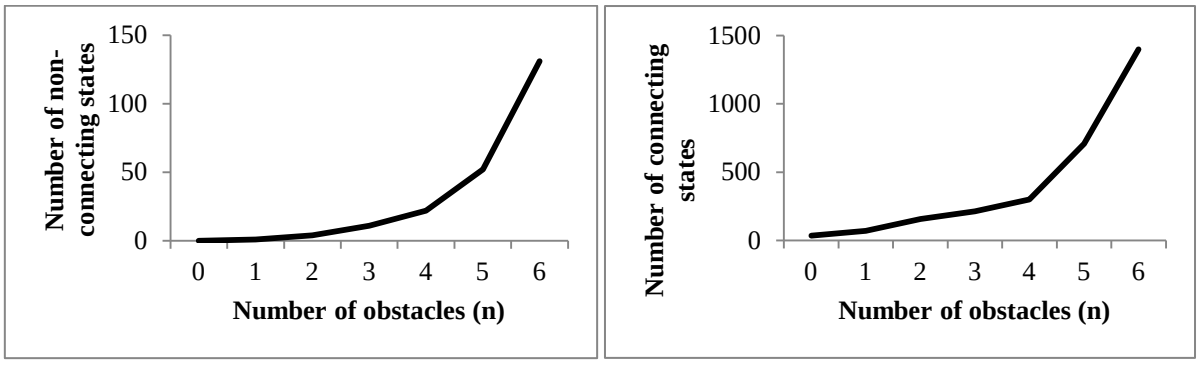

Figure 16: Increase in number of (a) non-connecting states and (b) connecting states with increase in number of obstacles

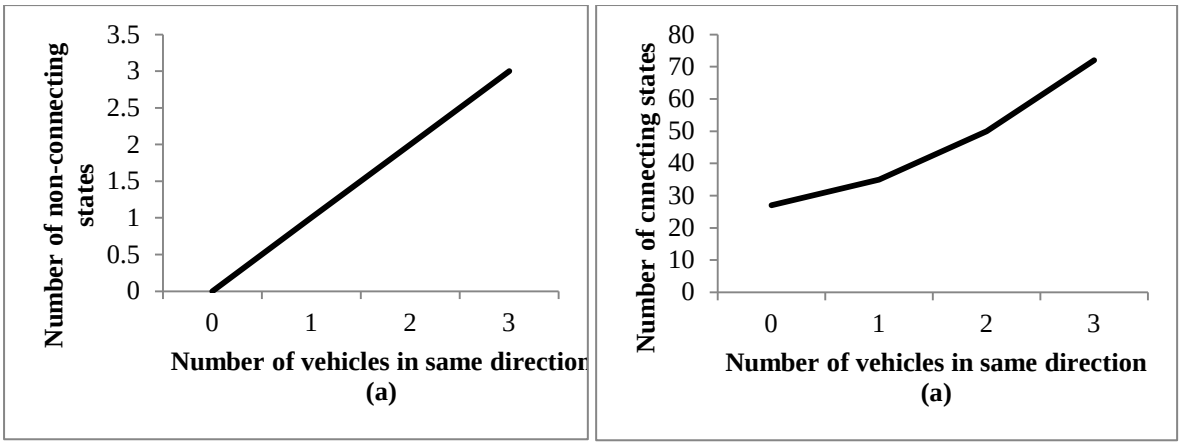

Figure 17: Increase in number of (a) non-connecting states and (b) connecting states for increasing number of vehicles in the same direction

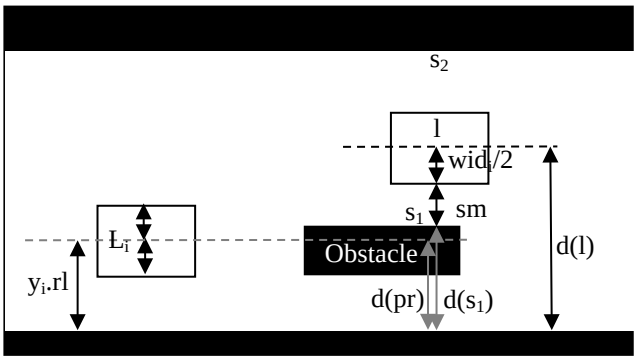

Figure 2: Computing state for expansion

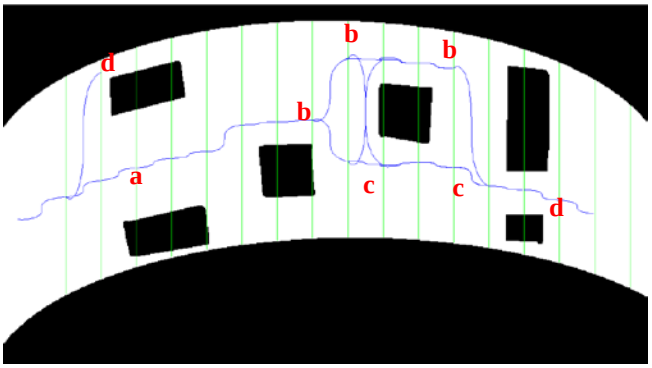

Figure 3: Various cases in selection of state for expansion

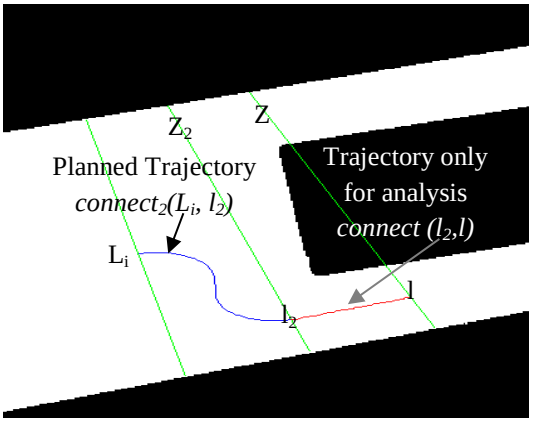

Figure 4: Object analysis and state expansion

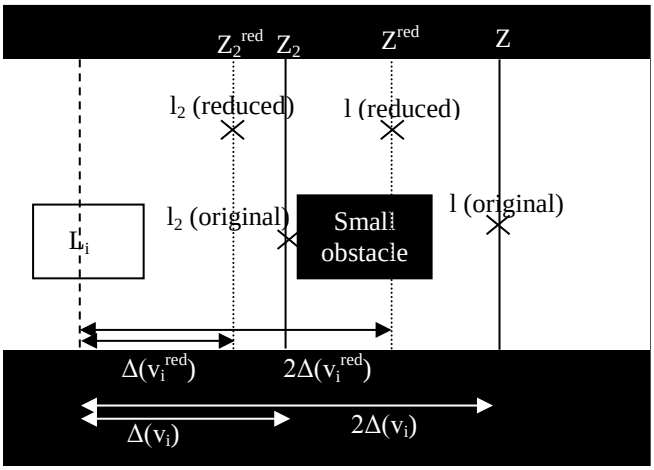

Figure 5: Small Obstacle Problem

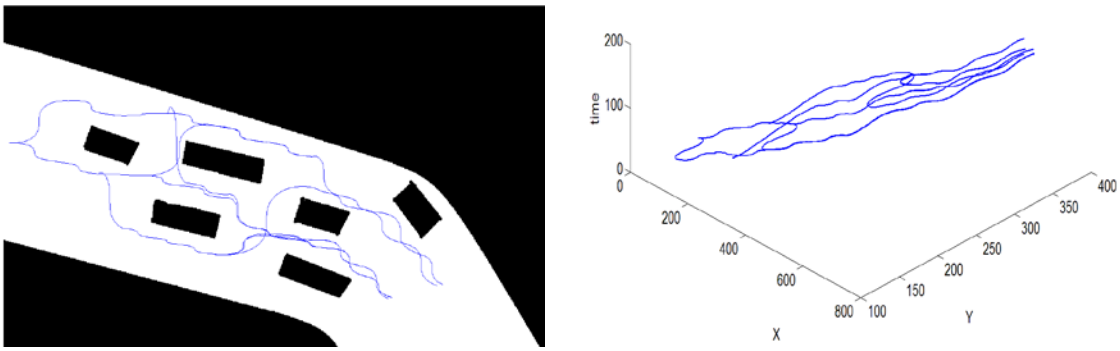

Figure 6: Graph generated for a sample problem (a) for given map (b) showing time axis

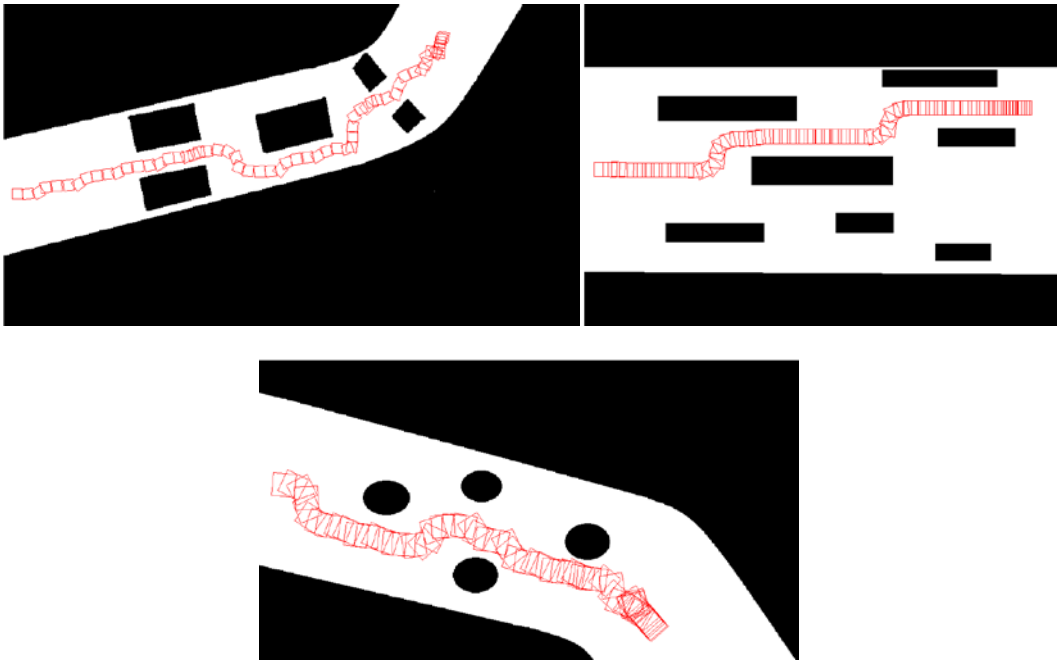

Figure 7: Experimental results with single vehicle

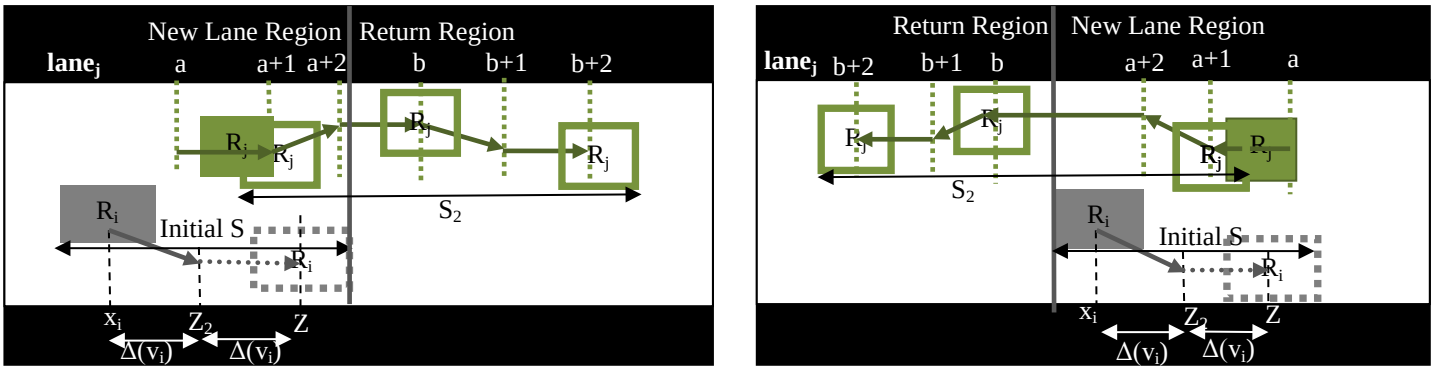

Figure 8: Calculating region of independent lanes (a) same direction of travel (b) opposite direction of travel

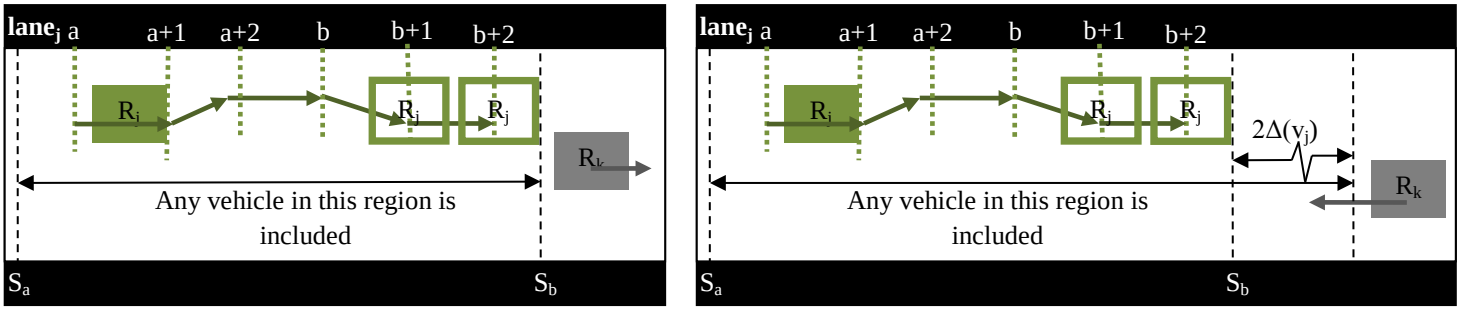

Figure 9: Decision regarding inclusion of R_k in H (a) same direction of travel (b) opposite direction of travel

Figure Captions

Figure 17: Increase in number of (a) non-connecting states and (b) connecting states for increasing number of vehicles in the same direction